WHAT IS CLAIMED IS:

1. A method performed on a computer system having a primary storage device, a secondary storage device, a display device, and an input/output mechanism which enables a client to dynamically distribute to a server computer in a collection of server computers a task developed in a programming language compatible with each of the server computers, the method comprising the steps of:

selecting a server from a plurality of heterogenous servers to process a task based upon the overall processing load distribution among the collection of server computers and the specialized computing capabilities of each server computer;

marshalling parameters and data into a task request which further comprises the substeps of,

determining if code and data types related to the requested task are present on the selected server, and

downloading the code and related data types onto the selected server when the code or data types are not present on the selected server;

invoking a generic compute method associated with the selected server which executes the task and further comprises the substeps of,

providing the task as a parameter to the generic compute method, and

indicating to the server that results from a computed task should be stored in a result cache on the selected server for subsequent tasks to use; and

receiving the computed task back from the selected server for further processing on the client.

2. A method performed on a processor contained within a computer system having a primary storage device, a secondary storage device, a display device, and an input/output mechanism which enables a server associated with a collection of servers to dynamically receive and process a task from a client computer wherein the task is in an executable programming language compatible with each of the server computers, the method comprising the steps of:

unmarshalling parameters and data from a task request into a task, which further comprises the substeps of,

determining if the types related to the task are available on the server, and

when the types related to the task are not available on the server, downloading the types onto the server from a location as indicated by the parameters provided by the client;

invoking a generic compute method, which is capable of processing all types of tasks, which executes the task and generates results;

storing results from the executed tasks in a cache if a subsequent task will use the results; and

returning the results from executed task to the client.

3. A method performed on a processor operatively coupled to a collection of servers which enables a client associated with the processor to dynamically distribute a task to a server, the method comprising the steps of:

selecting a server to process the task;

forming a task request from the parameters and data;

sending the task request to the selected server which invokes a generic compute technique capable of executing the task request on the selected server and generates results; and

receiving the results back from the selected server.

4. The method of claim 3, wherein the processor is operatively coupled to a computer system having a primary storage device, a secondary storage device, a display device, and an input/output mechanism.

5. The method of claim 3, wherein the task is developed in a programming language and environment compatible with each of the server computers.

6. The method of claim 3, wherein the server is selected from a plurality of heterogenous computer systems.

7. The method of claim 5, wherein the environment includes a remote procedure call subsystem.

8. The method of claim 7, wherein the remote procedure call subsystem is the Remote Method Invocation (RMI) system.

9. The method of claim 3, wherein a criteria for selection the server includes the overall processing load distribution among the collection of server computers.

10. The method of claim 6, wherein the selected server has the lowest load characteristic compared with average load characteristic of the servers over a predetermined time period.

11. The method of claim 3, wherein a criteria for selection the server includes the specialized computing capabilities of each server computer.

12. The method of claim 11, wherein the specialized computing capabilities includes rendering images.

13. The method of claim 3, wherein the sending step further comprises the substeps of:

determining if code related to the requested task is present on the selected server; and

downloading the code onto the selected server when the code is not present on the selected server.

14. The method of claim 3, wherein the sending step further comprises,

providing the task as a parameter to the generic compute method.

15. The method of claim 3 further comprising the step of indicating to the server that results from a computed task should be stored in a result cache on the selected server for subsequent tasks to use.

16. The method of claim 3, wherein the results are used for further processing on the client.

17. The method of claim 3, wherein the result is an object.

18. A method performed on a processor operatively coupled to a collection of servers which enables a server associated with the processor to dynamically receive and process a task from a client computer wherein the task is in an executable

programming language compatible with each of the server computers, the method comprising the steps of:

retrieving parameters and data from a task request into a task;

invoking a generic compute method on the server, which is capable of processing a plurality of types of tasks, which executes the task and generates results;

returning results to the client.

19. The method of claim 18, wherein the processor is operatively coupled to a computer system having a primary storage device, a secondary storage device, a display device, and an input/output mechanism.

20. The method of claim 18, wherein the task is developed in a programming language compatible with each of the server computers.

21. The method of claim 18, wherein the task is developed using the Java programming language and environment.

22. The method of claim 21, wherein the environment includes a remote procedure call subsystem.

23. The method of claim 22, wherein the remote procedure call subsystem is the Remote Method Invocation (RMI) system.

24. The method of claim 18, wherein the retrieving step further comprises,

determining if types related to the task are available on the server;

when the types are not available on the server, downloading the types onto the server from a location as indicated by the parameters provided by the client; and

executing the task based upon the data and parameters provided by the client.

25. The method of claim 24, wherein the determining step and the downloading steps are performed by a remote procedure call (RPC) subsystem.

26. The method of claim 25, wherein the determining step is performed by a Remote Method Invocation (RMI) type of remote procedure call subsystem.

27. The method of claim 18, further comprising the substep of storing the results from the task in a cache if a subsequent task will use the results.